



alfamacursos.com.br

A decorative orange wave graphic that curves upwards from the bottom right corner of the page.

Alfama Cursos

Antônio Garcez
Fábio Garcez
Diretores Geral

Antônio Álvaro de Carvalho
Diretor Acadêmico

MATERIAL DIDÁTICO

Produção Técnica e Acadêmica

Patrícia Queiroz de Meneses
Coordenadora Geral

Patrícia Queiroz de Meneses
Coordenadora Pedagógica

Neilton Barreto
Autoria

Gabriella Caroline Teles Silva
Sabina Regina Conceição Santos
Revisão Textual

Rafael Rezende de Farias
Editoração

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/98.
É proibida a reprodução total ou parcial, por quaisquer meios, sem
autorização prévia, por escrito, da
ALFAMA CURSOS.

alfamacursos.com.br



Banco de Dados II



Apresentação do Curso

A disciplina Banco de Dados é com certeza um dos mais importantes para a carreira do desenvolvedor, principalmente nos dias atuais, em que os paradigmas de desenvolvimento tradicional têm se modificado. Um bom programador deve conhecer e saber trabalhar com Sistemas de Gerenciamento de Banco de Dados para poder armazenar e resgatar as informações de seus sistemas. Imagine um sistema sem que possa armazenar informação? Nos dias atuais, isto é impraticável, logo esta disciplina é primordial para um profissional que queira se destacar no mercado, pois você vai aprender a utilizar um Sistema de Gerenciamento de Banco de Dados de forma prática e direta.

Apresentação do Professor

NEILTON BARRETO é bacharel em Tecnologia em Processamento de Dados, licenciado em Educação Profissional e pós-graduado em Engenharia de Sistemas, Gestão de Redes e Gestão em EAD. Atua na área de Informática e em cursos de qualificação profissional, como Cabeamento Estruturado, Servidores de Rede, Manutenção de Computadores e Impressoras.

Componente Curricular

EMENTA:

Introdução ao PHPMyAdmin; Criando Tabelas no PHPMyAdmin; Alterando Tabelas com o PHPMyAdmin; Inserindo Dados nas Tabelas; Exibindo Informações; MySQL Workbench; Criando e Removendo Banco de Dados com o Workbench; Criando Tabelas com o Workbench; Inserindo e Selecionando Dados na Tabela; Alterando e Removendo Tabelas; Modelagem de Dados; MySQL Workbench Revisão Geral; Criando Usuários; Administrando Usuários no MySQL; Backup do Banco de Dados; Chaves; Cardinalidade; Regras de Integridade de Dados no MySQL; Manipulando Dados em Tabelas Relacionadas; Seleção de Dados; Seleção: ordenação, agrupamento, Funções de Agrupamento; Junção; e Junção na Prática.

COMPETÊNCIAS:

- Demonstrar conhecimentos referentes à elaboração de banco de dados aplicados no desenvolvimento de sistemas de computação.
- Identificar os tipos de Banco de dados.
- Reconhecer o SQL.

HABILIDADES:

- Aplicar conhecimentos referentes ao banco de dados no desenvolvimento de sistemas de computação.
- Utilizar técnica de aplicação do Banco de dados.
- Instalar o Banco de Dados.

PÚBLICO-ALVO:

Estudantes de informática, técnicos de diversas áreas, empresas e repartições públicas.

Índice

Capítulo 1 - Introdução ao PHPMyAdmin	9
1.1 - Introdução	9
1.2 - Exercício Proposto	11
Capítulo 2 - Criando Tabelas no PHPMyAdmin	12
2.1 - Introdução	12
2.2 - Exercício Proposto	15
Capítulo 3 - Alterando Tabelas com o PHPMyAdmin	16
3.1 - Introdução	16
3.2 - Exercício Proposto	18
Capítulo 4 - Inserindo Dados em Tabelas	19
4.1 - Introdução	19
4.2 - Exercício Proposto	21
Capítulo 5 - Exibindo Informações	22
5.1 - Introdução	22
5.2 - Exercício Proposto	24
Capítulo 6 - MySQL Workbench	25
6.1 - Introdução	25
6.2 - Exercício Proposto	27
Capítulo 7 - Criando e Removendo Banco de Dados com o Workbench	28
7.1 - Introdução	28
7.2 - Exercício Proposto	30
Capítulo 8 - Criando Tabelas com o Workbench	31
8.1 - Introdução	31
8.2 - Exercício Proposto	33
Capítulo 9 - Inserindo e Selecionando Dados na Tabela	34
9.1 - Introdução	34
9.2 - Exercício Proposto	36
Capítulo 10 - Alterando e Removendo Tabelas	37
10.1 - Introdução	37
10.2 - Exercício Proposto	39
Capítulo 11 - Modelagem de Dados	40
11.1 - Introdução	40
11.2 - Exercício Proposto	42
Capítulo 12 - MySQL Workbench Revisão Geral	43
12.1 - Introdução	43
12.2 - Exercício Proposto	45
Capítulo 13 - Criando Usuários	46
13.1 - Introdução	46
13.2 - Exercício Proposto	48
Capítulo 14 - Administrando Usuários no MySQL	49
14.1 - Introdução	49
14.2 - Exercício Proposto	51
Capítulo 15 - Backup de Banco de Dados	52
15.1 - Introdução	52
15.2 - Exercício Proposto	53
Capítulo 16 - Chaves	54
16.1 - Introdução	54
16.2 - Exercício Proposto	56
Capítulo 17 - Cardinalidade	57
17.1 - Introdução	57
17.2 - Exercício Proposto	58
Capítulo 18 - Regras de Integridade de Dados no MySQL	59
18.1 - Introdução	59
18.2 - Implementando Chave Primária	59
18.3 - Exercício Proposto	60

Capítulo 19 - Manipulando Dados em Tabelas Relacionadas	61
19.1 – Introdução	61
19.2 – Exercício Proposto	62
Capítulo 20 - Seleção de Dados	63
20.1 – Introdução	63
20.2 – Exercício Proposto	64
Capítulo 21 - Seleção: ordenação, agrupamento	65
21.1 – Introdução	65
22.2 – Exercício Proposto	66
Capítulo 22 - Funções de Agregação	67
22.1 – Introdução	67
22.2 – Exercício Proposto	68
Capítulo 23 - Junção	69
23.1 – Introdução	69
23.2 – Exercício Proposto	70
Capítulo 24 - Junção na Prática	71
24.1 - Junção Produto Cartesiano	71
24.2 - Junção Interna/Inner Join	71
24.3 - Junção Externa	71
24.4 - Left Outer Join	71
24.5 - Right Outer Join	71
24.6 – Exercício Proposto	72
Referências	73

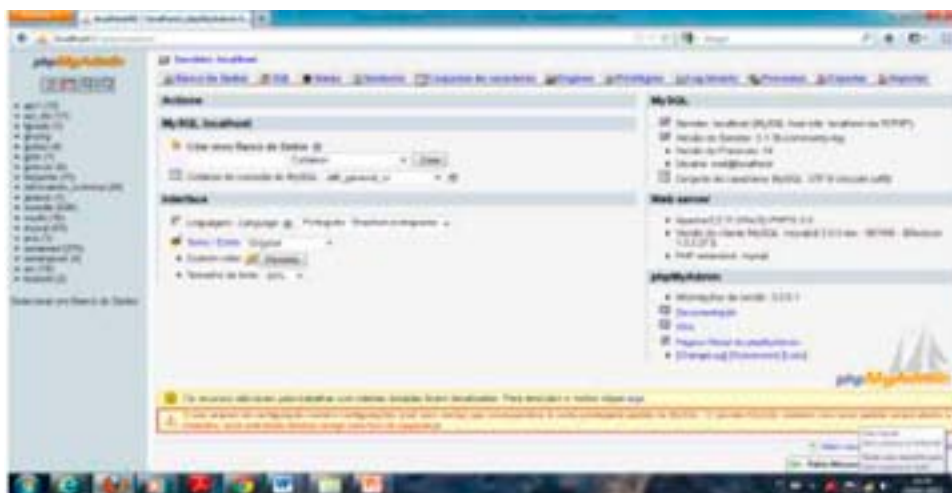
Capítulo 1 - Introdução ao PHPMyAdmin

1 - INTRODUÇÃO AO PHPMYADMIN

1.1- INTRODUÇÃO

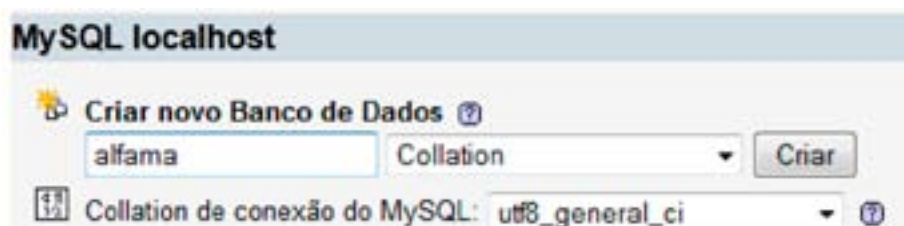
Um banco de dados, além de ser um elemento essencial para a gestão da informação é o núcleo de qualquer sistema de porte. Sistemas como o MySQL, Postgre, entre outros, ganham cada vez mais adeptos, principalmente no mundo da Web, em que o desempenho é o principal foco, mas há um grande problema, que é administrar um banco apenas utilizando linha de comando, além de não ser amigável, pode se tornar complexo e difícil. Outro ponto importante é que em um ambiente Web fazer acesso à linha de comando depende de protocolos específicos, o que pode dificultar a administração, logo a saída é a utilização de sistemas de gestão para ambientes Web. Para o MySQL o mais famoso e utilizado é o PHPMyAdmin, disponível em quase todos os servidores que dão suporte a PHP e MySQL devido a ser um ambiente de qualidade, fácil de utilizar e com muitos recursos. O PHPMyAdmin é uma aplicação para a administração de banco de dados MySQL. Como o PHPMyAdmin é multiplataforma, facilita o trabalho. Com o PHPMyAdmin é possível: criar, copiar, deletar, renomear tabelas e banco de dados, fazer manutenção, deletar, editar e adicionar campos às tabelas, além de possibilitar a exportação e a importação de dados e estrutura de banco.

A tela de abertura do MySQL é simples e fácil para acesso, como pode ser visto a seguir:



Note que entre os facilitadores, temos a criação de um novo banco de dados, normalmente, utilizaríamos o ambiente MySQL Command Line e com o comando Create Table, criaríamos nosso banco de dados. Para criar um novo banco de dados utilizando o PHPMyAdmin, no campo Criar novo Banco de Dados, insira o nome e clique no botão Criar.

Crie um banco de dados com o nome alfama, conforme exemplo abaixo:



Ao criar o novo banco, a mensagem abaixo será exibida e ao lado esquerdo da tela deve aparecer o banco de dados criado:

Database alfama has been created.



Apesar de facilitar a criação, ainda é possível criar banco de dados, tabelas ou efetuar consultas por comando, para isso, clique em SQL, que uma janela será aberta como na figura abaixo:



Insira o comando SQL e clique em Executar. O ambiente PHPMyAdmin deve executar o comando e exibir o resultado na tela, facilitando assim a administração e provendo um poderoso ambiente, que além de tornar mais fácil, dá poder ao desenvolvedor em momentos que a interface torna-se limitada.

Logo, o PHPMyAdmin é um ambiente de gerenciamento de banco de dados, destinado ao MySQL, conforme pode ser visto, ele torna a administração do banco mais simples e clara, além de tornar mais rápido o desenvolvimento.

E como obter o PHPMyAdmin?

No site oficial do PHPMyAdmin é possível fazer o download da versão completa:

http://www.phpmyadmin.net/home_page/index.php

Caso utilize ambiente de servidor como Wamp, Xampp, o PHPMyAdmin estará disponível também.

Para instalar no Linux(Debian) utilize;

`Apt-get install phpmyadmin`

Recordando

O PHPMyAdmin é um ambiente feito para o ambiente Web e proporciona maior facilidade na administração do banco de dados MySQL.

1.2 – EXERCÍCIO PROPOSTO

1) O que é o ambiente PHPMyAdmin?

Capítulo 2 - Criando Tabelas no PHPMyAdmin

2 - CRIANDO TABELAS NO PHPMYADMIN

2.1 – INTRODUÇÃO

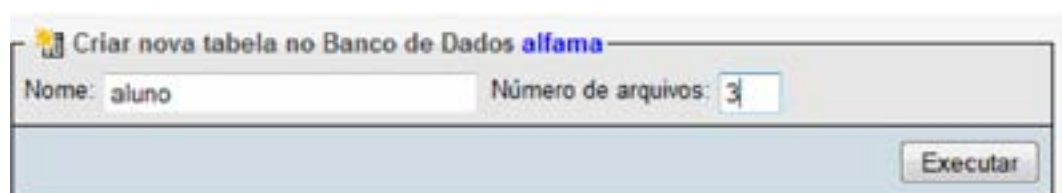
O PHPMyAdmin como software de administração de bancos de dados, permite a manipulação e a criação de bancos de dados, tabelas, etc.

Para a criação de tabelas no PHPMyAdmin, primeiro é necessário saber a quantidade de campos da tabela e o nome da tabela.



Para criar uma tabela é necessário o nome da tabela e a quantidade de campos que a tabela deve conter, por exemplo: tabela-aluno com os campos matrícula, nome, endereço, etc.

Vamos criar uma tabela-aluno com 3 campos.



Ao clicar em Executar, o PHPMyAdmin abre a tela para manipulação dos campos.



Na tela temos:

Campo = nome do campo.

Tipo = tipo do campo (int, varchar, etc.).

Tamanho = tamanho de dados aceitos pelo campo.

Nulo = permitir campo nulo.

Índice = tipo (primário, etc.).

AUTO_INCREMENT = habilita para fazer um campo autoincrementável.

Storage engine = tipo de engine a ser utilizável.

Praticando, crie os campos:

Id (int, chave primária e auto increment).

Login (varchar com 15 de tamanho).

Senha (varchar com 10 de tamanho).

Clique em Salvar.

Campo	id	login	senha
Tipo	INT	VARCHAR	VARCHAR
Tamanho/Definir ¹		15	10
Padrão ²	None	None	None
Collation			
Atributos			
Nulo	☐	☐	☐
Índice	PRIMARY		
AUTO_INCREMENT	☒	☐	☐
Comentários			

Comentários da tabela:
 Storage Engine: MyISAM
 Collation:

A tela pode ser exibida de duas formas, quando a poucos campos, temos a divisão horizontal como visto na figura acima, e quando temos mais de 4 campos a divisão vertical, conforme pode ser visto na figura abaixo.



Nome do campo, Tipo do campo, Tamanho do campo, Extra, pq?, Primária

- O resultado é a criação de uma tabela, seria o mesmo que executar o seguinte comando:

```
CREATE TABLE `alfama`.`aluno` ( `int` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
`login` VARCHAR( 15 ) NOT NULL, `senha` VARCHAR( 10 ) NOT NULL ) ENGINE = MYISAM ;
```

O comando pode ser executado via SQL para verificar o resultado. Ao ser executado aparecerá a mensagem de que foi criado com sucesso, como pode ser observado na figura abaixo.

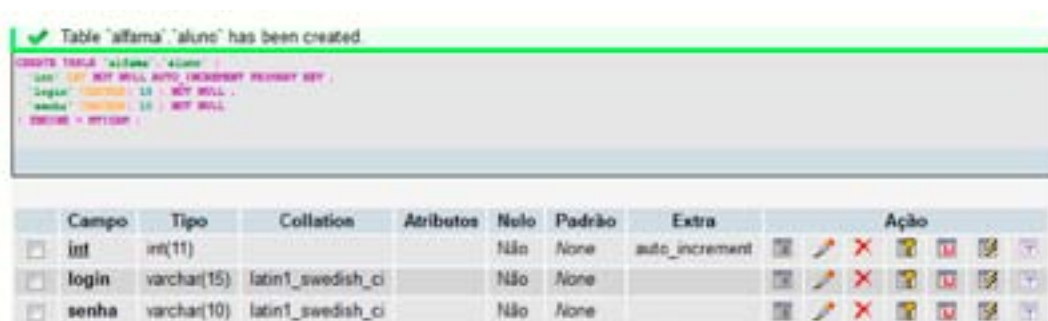


Table "alfama" "aluno" has been created.

Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ação
int	int(11)			Não	None	auto_increment	
login	varchar(15)	latin1_swedish_ci		Não	None		
senha	varchar(10)	latin1_swedish_ci		Não	None		

O PHPMyAdmin, além de facilitar a criação do banco de dados, permite que criemos tabelas. Outro ponto importante é que é fácil fazer um backup do banco, utilizando Exportar na parte superior.

Basta clicar em Exportar e depois em Executar. O padrão é exportar para SQL, mas é possível exportar para PDF, planilha, etc.

Para exportar, clique em Exportar e a tela a seguir será exibida.



Também é possível criar uma tabela através de SQL, clique em SQL e vamos criar a tabela a seguir:

Campo	Tipo	Tamanho	
id	Int	---	Chave Primaria, auto_Increment, not null
Nome_usuario	Varchar	50	Not null
Login_usuario	Varchar	10	Not null
Senha_usuario	varchar	10	Not null

Para criar esta tabela-usuário faça o comando a seguir:

```
Script SQL
CREATE TABLE usuario (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  nome_usuario VARCHAR (50) NOT NULL,
  login_usuario VARCHAR (10) NOT NULL,
  senha_usuario VARCHAR (10) NOT NULL,
  PRIMARY KEY('id')) TYPE = InnoDB ;
```

Pronto! A tabela-usuário foi criada com sucesso.

Recordando

O PHPMyAdmin é um ambiente integrado que possibilita a manipulação e a gestão de bancos de dados MySQL, também nota-se a facilidade na criação de tabelas, como vimos anteriormente.

2.2 – EXERCÍCIO PROPOSTO

1) É possível criar uma tabela através do comando SQL no ambiente PHPMyAdmin?

Capítulo 3 - Alterando Tabelas com o PHPMyAdmin

3 - ALTERANDO TABELAS COM O PHPMYADMIN

3.1 – INTRODUÇÃO

Com o PHPMyAdmin é possível modificar as tabelas adicionando novos campos, removendo campos desnecessários, alterando os tipos de campos, entre outros.

Como apresentado na figura, indique a quantidade de campos a adicionar, e a localização (início ou fim da tabela) e clique em Executar.



Na tabela, caso queira inserir o campo log para armazenar os logs de usuário, clique em Executar (Note que já tem o adicionar 1 campo no final da tabela). Adicione um campo log com o tipo int (inteiro) e salve.

Campo	log
Tipo	INT
Tamanho/Definir ¹	
Padrão ²	None
Collation	
Atributos	
Nulo	☐
Índice	---
AUTO_INCREMENT	☐
Comentários	

Salvar Ou Adicionar 1 campo(s) Executar

O PHPMyAdmin está na verdade executando o comando Alter Table com a função de adicionar campo. Quando o administrador clicar em Salvar, o resultado será a criação do novo campo, conforme pode-se observar na figura abaixo:



Caso seja necessário, é possível alterar o tipo de um campo. Vamos alterar o campo que foi criado anteriormente, do tipo int para o tipo varchar. Clique em editar conforme indicado na figura abaixo:

Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ação
☐ int	int(11)			Não	None	auto_increment	    
☐ login	varchar(15)	latin1_swedish_ci		Não	None		    
☐ senha	varchar(10)	latin1_swedish_ci		Não	None		    
☐ log	int(11)			Não	None		    

Clique aqui para editar.

Na nova tela que será exibida troque o tipo do campo, conforme apresentado na figura a seguir:

Campo	log
Tipo	INT
Tamanho/Definir ¹	VARCHAR
Padrão ²	TEXT
Collation	DATE
Atributos	NUMERIC
Nulo	TINYINT
AUTO_INCREMENT	SMALLINT
Comentários	MEDIUMINT
	INT
	BIGINT
	-
	DECIMAL
	FLOAT
	DOUBLE
	REAL
	-
	BIT
	BOOL
	SERIAL
	DATE and TIME

¹ Se um tipo de campo for selecionado, o tamanho será definido automaticamente.
² Se você for colocar um padrão, o campo será definido como NOT NULL.

É possível alterar o tipo, tamanho, etc.

Caso queira, é possível ainda excluir um campo clicando no X, conforme podemos ver abaixo.

Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ação
☐ int	int(11)			Não	None	auto_increment	    
☐ login	varchar(15)	latin1_swedish_ci		Não	None		    
☐ senha	varchar(10)	latin1_swedish_ci		Não	None		    
☐ log	int(11)			Não	None		    

Clique aqui para excluir.

Recordando

Como vimos anteriormente, o PHPMyAdmin além de tornar fácil a manipulação de bancos de dados, torna fácil a modificação do banco.

3.2 – EXERCÍCIO PROPOSTO

1) Quais são respectivamente os desenhos descritos no ambiente PHPMyAdmin para ALTERAÇÃO e EXCLUSÃO?

Capítulo 4 - Inserindo Dados em Tabelas

4 - INSERINDO DADOS EM TABELAS

4.1 – INTRODUÇÃO

Outro ponto importante é que o PHPMyAdmin possibilita que o administrador insira dados nas tabelas.

Para inserir dados, clique em Inserir, conforme a figura abaixo.

Depois de clicar em inserir, insira os dados e clique no botão Executar, lembre-se de que o campo auto_increment não precisa ser preenchido.

O PHPMyAdmin exibe o comando e apresenta que foi inserido 1 row (uma linha) com sucesso).

Também é possível inserir dados no PHPMyAdmin através dos comandos SQL.

Clique em SQL e insira o código Insert.

Depois de inserir o comando, clique em Executar.

```
INSERT INTO `alfama`.`aluno` (`int`, `login`, `senha`, `log`) VALUES (NULL, 'fabio', 'fabio', 'teste');
```



O comando Insert possui a seguinte sintaxe:

```
INSERT [INTO] nome_tabela (colunas) VALUES (valores)
```

Onde:

- **nome_tabela:** é o nome da tabela que deseja incluir os dados.
- **colunas:** é o nome das colunas da tabela que deseja acrescentar os dados.
- **valores:** é o conteúdo de cada coluna.

Vamos apresentar um exemplo inserindo dados em uma tabela-usuário, que possua os campos nome_usuario, login_usuario e senha_usuario.

```
INSERT INTO usuário (nome_usuario, login_usuario, senha_usuario) values ('fabio', 'teste', '123456');
```

Neste caso, estamos inserindo dados na tabela-usuário, campos nome_usuario, login_usuario e senha_usuario, os valores estão em values, quando os dados são caracteres, utilizamos aspas simples para a inserção.

4.2 – EXERCÍCIO PROPOSTO

1) Quando executamos o comando abaixo no SQL do PHPMyAdmin resulta em que?
`INSERT INTO `alfama`.`aluno` (`int`, `login`, `senha`) VALUES (NULL, `fabio`, `fabio.123`)`

Capítulo 5 - Exibindo Informações

5 - EXIBINDO INFORMAÇÕES

5.1 – INTRODUÇÃO

O PHPMyAdmin é intuitivo quando se trata de pesquisa, além de facilitar a exibição dos dados, ele possibilita fazer consultas manuais, como veremos adiante.

Clique na tabela e será possível visualizar os dados já cadastrados.

			int	login	senha	log
☐			1	neilton	neilton@12	0
☐			2	Neilton	neilton@12	0
☐			3	teste	teste	0
☐			4	fabio	fabio	0

Aqui é possível ainda remover um registro, para isso clique sobre o X.

				int	login	senha	log
				1	neilton	neilton@12	0
				2	Neilton	neilton@12	0
				3	teste	teste	0
				4	fabio	fabio	0

Podemos ainda modificar dados, por exemplo, a senha, clicando sobre o lápis.

				int	login	senha	log
				1	neilton	neilton@12	0
				2	Neilton	neilton@12	0
				3	teste	teste	0
				4	fabio	fabio	0

Com o PHPMyAdmin é possível limitar a quantidade de informações que serão exibidas, visualizar para impressão, exportar, etc.

Mostrar 30 registro(s) começando de 0

no modo horizontal e repetindo cabeçalhos após 100 células

Operações resultantes das consultas

Visualização para impressão Ver impressão (com textos completos) Exportar CREATE VIEW

Se a tabela tiver mais de 30 registros por padrão, o PHPMyAdmin apresenta os primeiros trinta. Veja o exemplo:

```

Mostrando registros 0 - 29 (~3621 total, Consulta levou 0.0004 segundos)

SELECT *
FROM 'edi_capabilizier'
LIMIT 0, 30
    
```

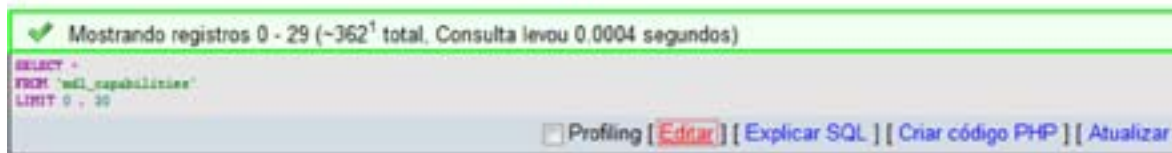
Depois de alterar a quantidade clique em > para avançar, conforme a figura abaixo.



Podemos alterar a quantidade de registros que serão apresentados, alterando o campo de registros e clicando em Mostrar.



É possível ainda alterar os campos que serão exibidos, clique em Editar.



Troque o * pelo nome dos campos, clicando sobre o campo e na seta <<. Coloque em cada campo. Clique em Executar.



O resultado é que ao invés de exibir todos os campos, o PhpMyAdmin vai exibir apenas os campos que foram solicitados.

			name	component
☐			moodle/site:config	moodle
☐			moodle/site:readallmessages	moodle
☐			moodle/site:sendmessage	moodle

5.2 – EXERCÍCIO PROPOSTO

1) O processo de exibição das informações no ambiente PHPMyAdmin é realizado todo no modo gráfico? Por quê?

Capítulo 6 - MySQL Workbench

6 - MYSQL WORKBENCH

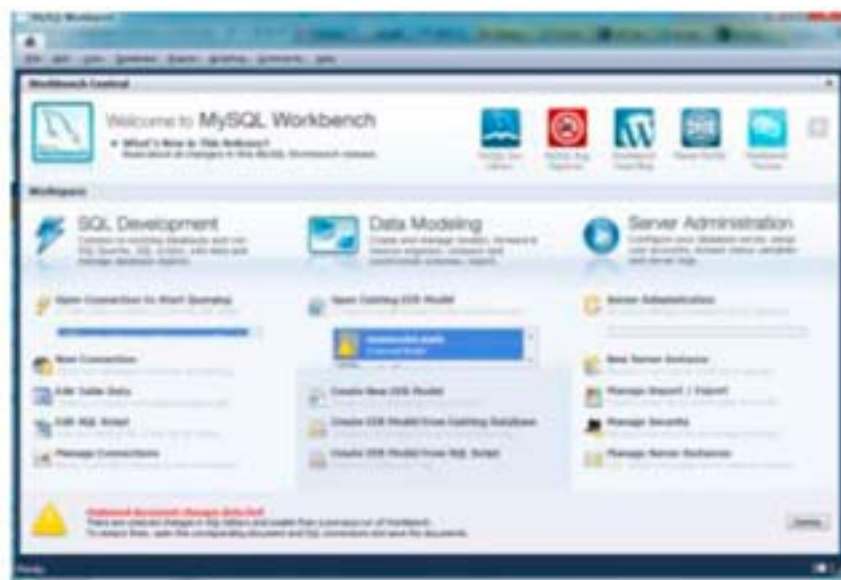
6.1 – INTRODUÇÃO

O MySQL Workbench é um software com versão grátis e comercial, que possibilita administrar bancos de dados MySQL, assim como o PHPMyAdmin, com a principal diferença de ser mais robusto, mas não funcionar em uma plataforma Web como o PHPMyAdmin.

A versão atual do MySQL Workbench é a 5.2 e pode ser feito download gratuitamente da versão CE (Community Edition) em:

<http://www.mysql.com/products/community/>

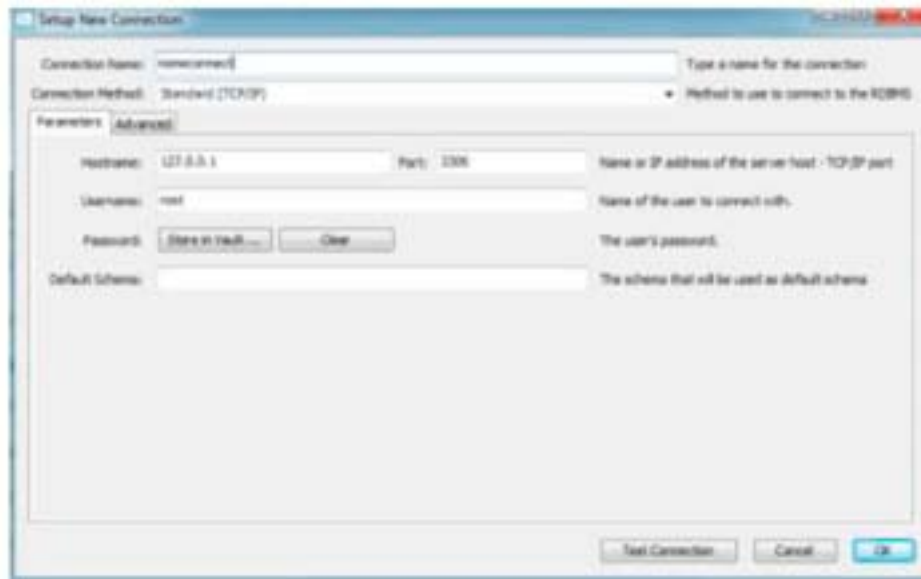
Depois de instalado, e ao abrir, a primeira tela que será exibida será:



A primeira tarefa é criar uma conexão com o seu servidor MySQL.



Coloque o nome e o caminho do servidor e clique em OK.



Depois de configurado, para utilizar a conexão basta clicar sobre a conexão duas vezes, que o Workbench vai abrir o seu banco de dados.



O MySQL Workbench foi criado para ser intuitivo, além de possibilitar uma boa administração dos dados.

Permite que você trabalhe com diversos servidores instalados, internamente e remotamente. Na primeira tela onde criamos nossa conexão existem outras possibilidades.

- **New Connection:** permite adicionar uma nova conexão de banco de dados para consulta.
- **Create New EER Model:** permite criar um modelo de diagrama relacional em branco.

6.2 – EXERCÍCIO PROPOSTO

1) Qual vantagem e desvantagem do MySQL Workbench em relação ao PHPMyAdmin?

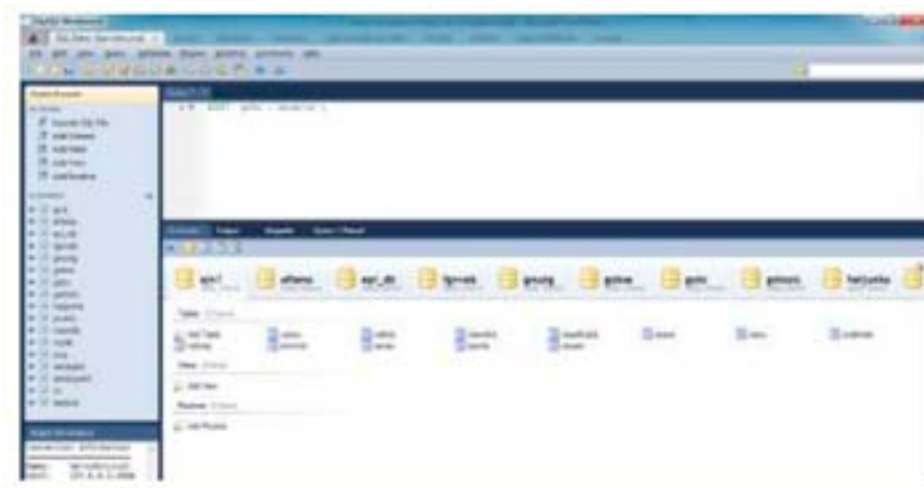
Capítulo 7 - Criando e Removendo Banco de Dados com o Workbench

7 - CRIANDO E REMOVENDO BANCO DE DADOS COM O WORKBENCH

7.1 – INTRODUÇÃO

Assim como o PHPMyAdmin, o Workbench permite que se crie, remova e manipule bancos de dados MySQL.

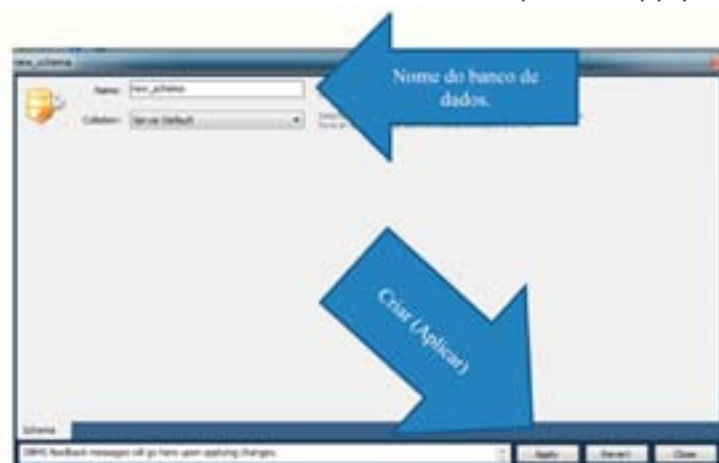
Após realizada uma conexão, a tela do Workbench será:



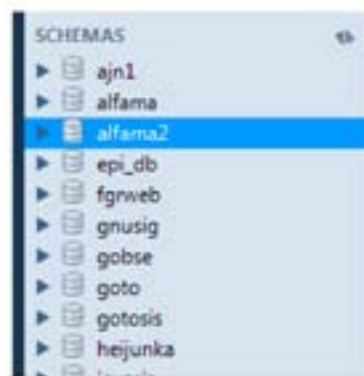
No lado esquerdo temos o Object Browser. Para adicionar um novo banco clique em Add Schema:



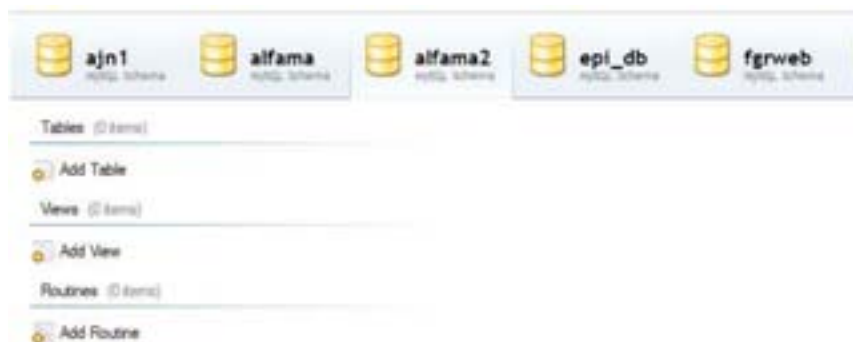
Na tela de new_schema adicione o nome do banco e clique em Apply.



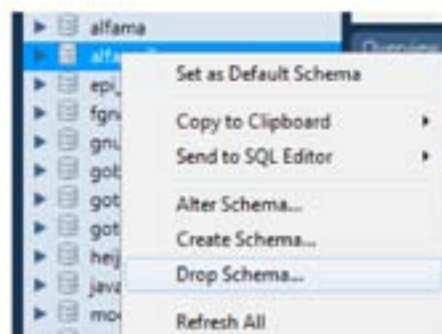
Será exibida uma nova tela com o comando SQL. Clique em Apply novamente para que o Workbench envie o comando para o servidor MySQL, depois clique em Finish. Todos os bancos de dados são exibidos no lado esquerdo, em SCHEMAS.



Também é possível visualizar na tela central.



Para remover um banco de dados, clique sobre o nome do SCHEMA com o botão direito e clique em Drop Schema, é o mesmo que executar o comando Drop.



O Workbench vai abrir uma tela de confirmação, clique em Apply e em seguida em Finish. Pronto! O banco de dados foi removido do seu servidor, mas lembre-se, é uma tarefa sem volta, logo, para remover, tenha certeza de que o banco não é mais necessário, caso contrário faça um backup anteriormente.

Recordando

Com o Workbench é possível fazer as mesmas tarefas que são feitas por linha de comando ou com o PHPMyAdmin, mas sendo um ambiente mais intuitivo e prático para a administração de banco de dados.

7.2 – EXERCÍCIO PROPOSTO

1) Para que serve a função Add Schema no MySQL Workbench?

Capítulo 8 - Criando Tabelas com o Workbench

8 - CRIANDO TABELAS COM O WORKBENCH

8.1 – INTRODUÇÃO

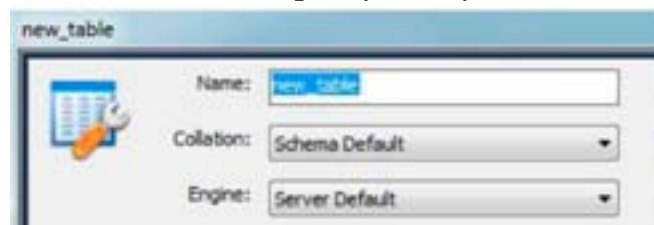
O MySQL Workbench possui mais recursos que o PHPMyAdmin para a criação e manipulação de bancos de dados MySQL, conforme já vimos anteriormente.

Vamos utilizar o Workbench para criar tabela.

Clique em Add Table para adicionar tabela.



Coloque o nome da tabela e escolha o engine (innodb).



Clique em Columns para adicionar campos à tabela.



Column Name = nome campo.

Datatype = tipo do campo.

PK = chave primária.

NN = not null.

AI = auto_increment.

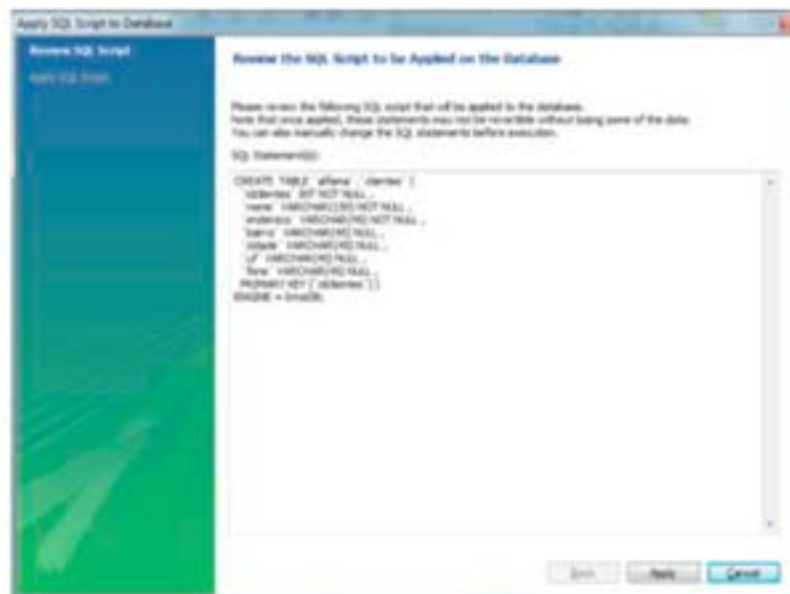
Column Name	Datatype	PK	NN	UQ	BN	UN	ZF	AI	Default
idclientes	INT	☒	☒	☐	☐	☐	☐	☐	

Vamos criar uma tabela-clientes, com os campos:

Idclientes (inteiro, chave primária e auto increment)
 Nome (varchar com 150 de tamanho)
 Endereço
 Bairro
 Cidade

Column Name	Datatype	PK	NN	UQ	BN	UN	ZF	AI	Default
Idclientes	INT	☒	☒	☐	☐	☐	☐	☐	
nome	VARCHAR(150)	☐	☒	☐	☐	☐	☐	☐	
endereço	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	
bairro	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	
cidade	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	
uf	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	
fone	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	

Após indicar o nome da tabela e os campos que farão parte dela, clique no botão Apply. Uma nova tela será apresentada com os comandos SQL para criação da tabela. Clique em Apply e Finish.



Como foi feito no PHPMyAdmin, o Workbench também permite que se criem tabelas, bancos de dados e outros recursos através de comandos SQL.



8.2 – EXERCÍCIO PROPOSTO

1) Explique o processo de criação de tabelas no MySQL Workbench.

Capítulo 9 - Inserindo e Selecionando Dados na Tabela

9 - INSERINDO E SELECIONANDO DADOS NA TABELA

9.1 – INTRODUÇÃO

Para inserir dados em uma tabela, o workbench disponibiliza tanto o ambiente de inserção como o de comandos.

Para inserir pelo ambiente, clique duas vezes sobre a tabela.



Basta inserir os dados desejados.



Ao concluir, clique em Apply Changes.



O MySQL Workbench abre a tela de Apply com o comando, clique em Apply --> Finish.

Outra forma de inserir dados na tabela é na parte superior (Query), que roda o comando de Insert, por exemplo.

```
INSERT INTO `alfama`.`clientes` (`nome`, `endereco`, `bairro`, `cidade`, `uf`, `fone`) VALUES ('Fabio', 'teste', 'Salgado Filho', 'Aracaju', 'SE', '0000');
```

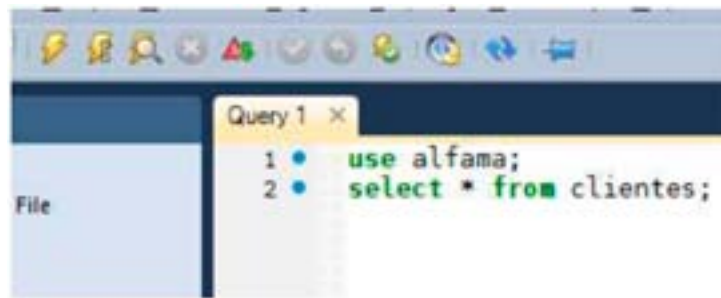
O resultado é o mesmo.

O comando Select serve para selecionar todos os registros ou uma sequência deles em uma tabela.

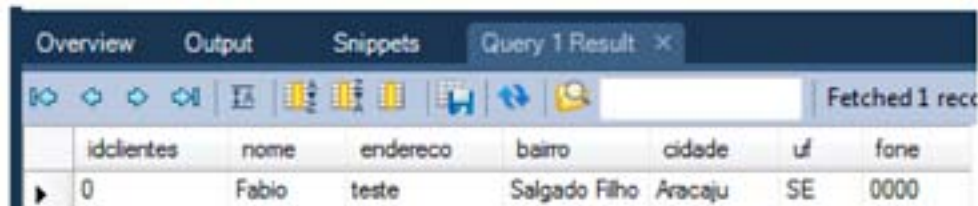
Select campo from tabela;

Vamos fazer um exemplo:

```
Select * from clientes
```



Em Query Result, podemos ver o resultado de nossa pesquisa.



	idclientes	nome	endereço	bairro	cidade	uf	fone
0		Fabio	teste	Salgado Filho	Aracaju	SE	0000

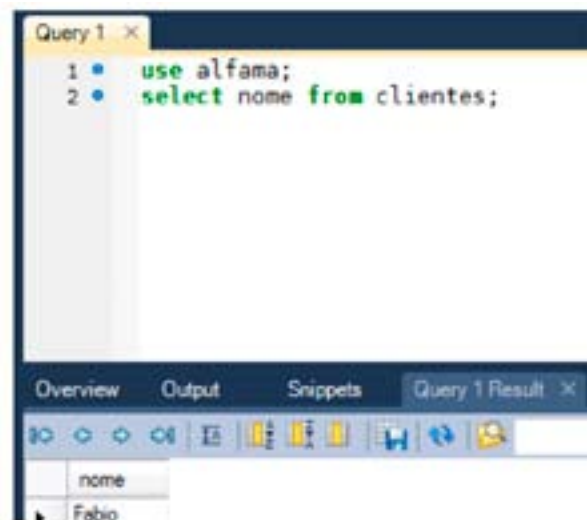
- **Campo:** os campos que desejo que sejam mostrados no resultado da minha consulta. Caso queira todos os campos, utilize o asterisco (*).
- **Tabela:** tabela a ser selecionada.

Existem ainda as opções Where, Order By, mas veremos isto mais à frente.

- `select * from clientes;`

Ele retorna todos os registros armazenados na tabela-clientes. Caso queira apenas exibir os nomes de usuários registrados na tabela-usuário, basta utilizar o nome do campo desejado, separando por vírgula quando for mais de um campo.

- `select nome from clientes;`



- `select * from clientes where idade > 18;`

Retorna todos os usuários que tenham o campo idade maior que 18.

- `select * from usuario Order by nome_usuario;`

Retorna todos os usuários ordenados pelo campo nome_usuario.

9.2 – EXERCÍCIO PROPOSTO

1) Explique o processo de inserção de dados em uma tabela no MySQL Workbench.

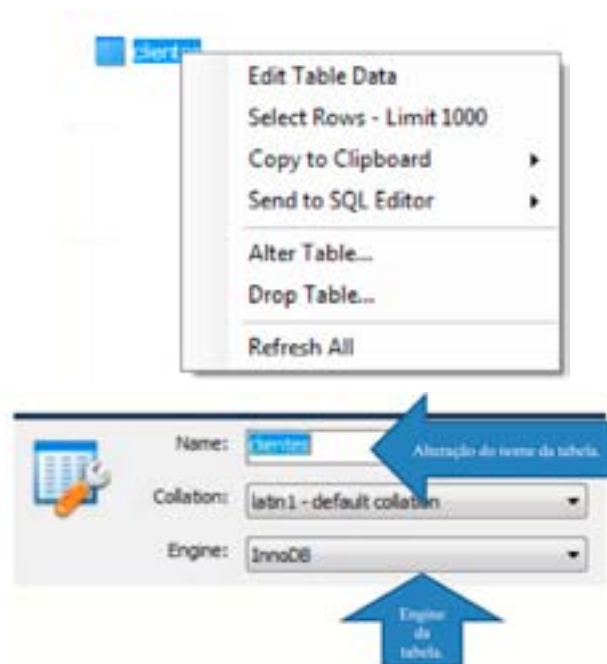
Capítulo 10 - Alterando e Removendo Tabelas

10 - ALTERANDO E REMOVENDO TABELAS

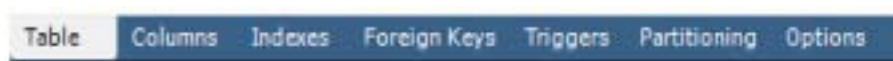
10.1 - INTRODUÇÃO

A alteração de tabelas é feita através do comando Alter Table, mas o Workbench possibilita uma alteração mais simples e prática.

Clique com o botão direito sobre a tabela a ser alterada e em Alter Table.



Na parte inferior temos: Columns que serve para alterar, inserir ou remover colunas; Indexes para inserir novos índices; Foreign Keys para chave estrangeira; Triggers para gatilhos (deve ser innodb o engine); Partitioning para partições do banco e Options para opções do banco.



Clique em Columns.



Para remover um campo, clique com o botão direito sobre o campo e em delete Selected Columns, para inserir basta colocar o campo aonde desejar, para alterar nome e tipo basta clicar no nome e fazer alterações.

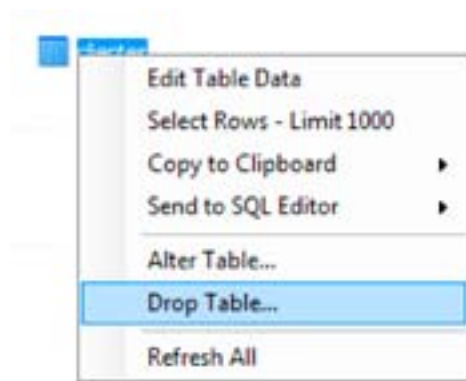
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idclientes	INT(11)	☒	☒	☐	☐	☐	☐	☐	
nome	VARCHAR(150)	☐	☒	☐	☐	☐	☐	☐	
endereco	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	
bairro	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
cidade	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
uf	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
fone	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL

Ao concluir basta clicar em Apply --> Apply e Finish.

As alterações foram bem-sucedidas. Outra forma de alterar é através do comando Alter Table, vamos fazer um exemplo de inserir o campo fone2.

```
ALTER TABLE `alfama`.`clientes` ADD COLUMN `fone2` VARCHAR(45) NULL AFTER `fone` ;
```

Para remover tabelas, podemos utilizar o comando Drop Table nome_tabela ou através do ambiente Workbench.



10.2 – EXERCÍCIO PROPOSTO

1) O que referencia o comando abaixo:

```
ALTER TABLE `alfama`.`clientes` ADD COLUMN `fone2` VARCHAR(45) NULL AFTER  
`fone`;
```

Capítulo 11 - Modelagem de Dados

11 - MODELAGEM DE DADOS

11.1 – INTRODUÇÃO

A modelagem de dados foi apresentada por Peter Chen na década de 70, foi um marco no desenvolvimento de sistemas de informação, criando uma mudança de paradigma para a criação dos bancos de dados.

A fase de modelagem é o momento em que o administrador de banco de dados, faz o projeto do banco de dados, isto é, o momento de estruturar a solução a ser implementada pelo sistema de gerenciamento de banco de dados. A modelagem de sistemas, tanto funcional quanto de dados faz parte da Engenharia de Requisitos, sendo de extrema importância para o sucesso de qualquer projeto, principalmente nos projetos de banco de dados.

A modelagem facilita o entendimento do contexto do negócio onde será inserida a aplicação. O modelo criado serve para visualização por parte do cliente e por parte dos desenvolvedores, facilitando a integração dos recursos.

Mas por que modelar?

A modelagem ajuda a entender as necessidades de informações para a criação das tabelas, ajuda a evitar redundância de informação e serve como projeto e documento do seu banco de dados.

Para modelar, devemos fazer a Engenharia de Requisitos e seguir algumas regras:

- Determinar qual a finalidade do banco de dados.
- Dividir o problema em várias tabelas.
- Determinar como as tabelas vão se relacionar.
- Refinar o modelo e cuidar com o tamanho dos campos.

Veja que é semelhante ao processo de modelagem de sistemas, claro, aplicado diretamente aos dados e ao SGBD.

Os modelos criados seguem um padrão de entidade e relacionamento, temos as entidades representadas pelos quadrados (retângulos) e os relacionamentos, representados por linhas.

O Workbench permite que seja feita a modelagem de dados, ou seja, uma forma mais simples e prática de se planejar seu banco de dados.

Como já criamos duas tabelas no banco de dados alfama, vamos pedir para o Workbench fazer a engenharia reversa e criar a modelagem com base no que nós já temos.

Na tela principal clique em Create EER Model From Existing Database.



Será exibida a tela para seleção do servidor. Clique em Next --> Next.



Selecione o banco de dados para criar o modelo ER --> Next --> Next --> Execute --> Next --> Finish.

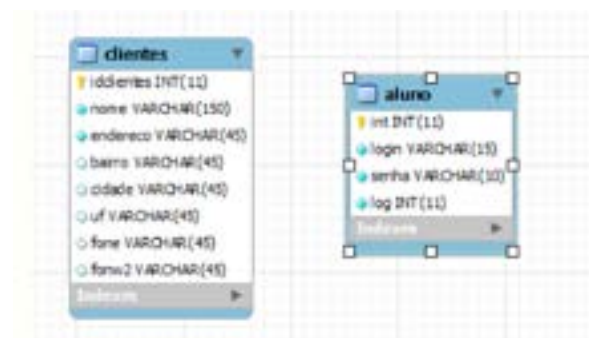
Select Schemata to Reverse Engineer



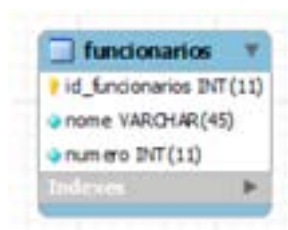
Select the Schemata to be Processed:

- ☐ agn1
- ☒ alfama
- ☐ epi_db
- ☐ fgweb
- ☐ ghuig
- ☐ gobse
- ☐ goto
- ☐ gotow

O resultado é o Diagrama de EER do banco de dados alfama que foi criado anteriormente.

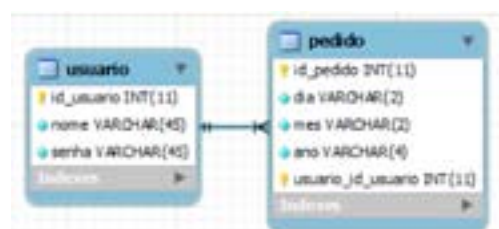


A modelagem ajuda o desenvolvedor a visualizar os dados que serão necessários para uma tabela, a representação feita com um quadrado, o título/nome da tabela que está no topo e os campos dentro do quadrado.



Os modelos são semelhantes às classes no UML.

Aqui temos um modelo tabela-usuário, campos id_usuario do tipo Int com 11 de tamanho, campo nome do tipo varchar com 45 de tamanho, etc.



11.2 – EXERCÍCIO PROPOSTO

1) Quais regras devemos seguir para fazermos uma modelagem de dados, diante da Engenharia de Requisitos?

Capítulo 12 - MySQL Workbench Revisão Geral

12 - MYSQL WORKBENCH REVISÃO GERAL

12.1 – INTRODUÇÃO

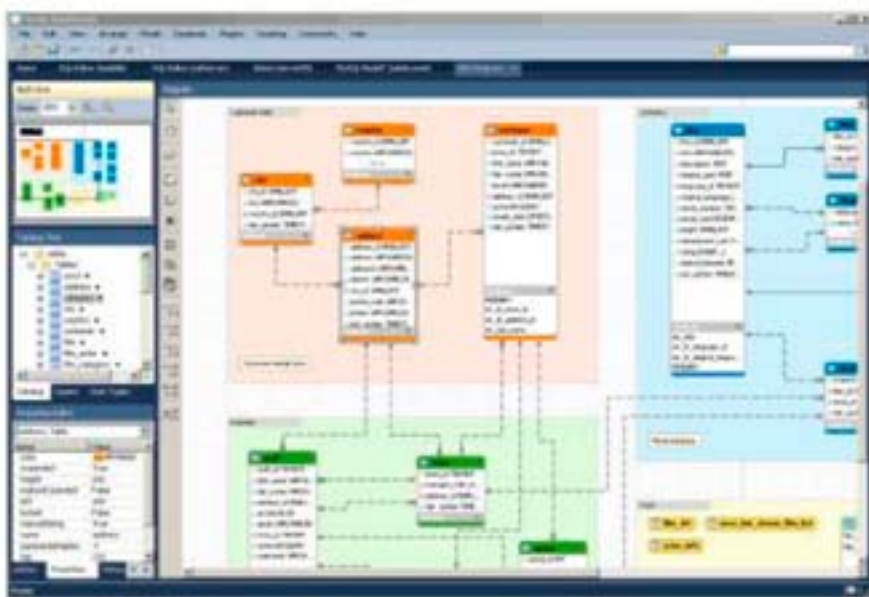
O MySQL Workbench é um software de modelagem e administração de banco de dados, disponível gratuitamente para o MySQL em:

<http://www.mysql.com/products/workbench/>

Após instalado, ao abrir o Workbench a seguinte tela será exibida:



O item Data Modeling permite criar modelos de banco de dados.



Clique em Create New EER Model.



Clique em Add Diagram.



Dê dois cliques na imagem table1 que foi adicionada, que irá aparecer a propriedade para colocar o nome da tabela.

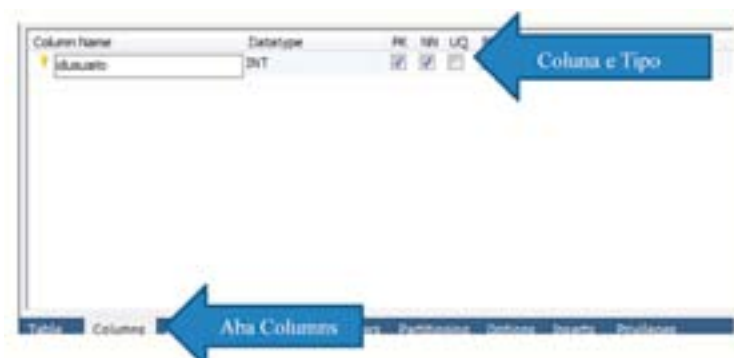


Clique na aba Columns na parte inferior da propriedade para adicionar colunas.



Com o Workbench é possível criar todo o modelo do banco de dados, com a vantagem dele gerar o código SQL para ser usado no banco de dados.

Para isso, depois de criado o modelo clique em File --> Export --> Forward SQL Create.



12.2 – EXERCÍCIO PROPOSTO

1) Com o Workbench é possível criar todo o modelo do banco de dados em código SQL para ser usado posteriormente?

Capítulo 13 - Criando Usuários

13 - CRIANDO USUÁRIOS

13.1 – INTRODUÇÃO

O MySQL é um Sistema de Gerenciamento de Banco de Dados multiusuário. Como todo Sistema de Gerenciamento de Banco de Dados de qualidade, o MySQL possui os recursos de usuários, que podemos adicionar usuários e dar permissões específicas.

Primeiramente, é necessário acessar a linha de comando ou utilizar o modo SQL de algum dos programas citados no capítulo anterior. Ao se logar como Administrador ou Root, a linha de comando do MySQL fica da seguinte forma:

Mysql>

Para criar um novo usuário utilizaremos o comando Grant.

Sintaxe do comando Grant:

GRANT priv [(colunas)] [, priv [(colunas)]] ...

ON {*. * | db.* | db.tabela}

TO usuario [IDENTIFIED BY 'senha']

[, usuario [IDENTIFIED BY 'senha']] ...

[WITH [GRANT OPTION |

MAX_QUERIES_PER_HOUR contador |

MAX_UPDATES_PER_HOUR contador |

MAX_CONNECTIONS_PER_HOUR contador]]

Privilégio	Descrição
ALL [PRIVILEGES]	Todos os privilégios exceto GRANT OPTION.
ALTER	Permite executar ALTER TABLE.
CREATE	Permite executar CREATE TABLE.
CREATE TEMPORARY TABLES	Permite executar CREATE TEMPORARY TABLE.
DELETE	Permite executar DELETE.
DROP	Permite executar DROP TABLE.
EXECUTE	Permite executar STORED PROCEDURES (MySQL 5.0).
FILE	Permite executar SELECT... INTO OUTFILE e LOAD DATA INFILE.
INDEX	Permite executar CREATE INDEX e DROP INDEX.
INSERT	Permite executar INSERT.
LOCK TABLES	Permite executar LOCK TABLES em tabelas que você tenha o privilégio SELECT.
PROCESS	Permite executar SHOW FULL PROCESSLIST.
REFERENCES	Ainda não está implementado.
RELOAD	Permite executar FLUSH.
REPLICATION CLIENT	Permite ao usuário obter a localização do Master ou Slave.
REPLICATION SLAVE	Necessário para a replicação Slave (leitura dos eventos do log binário do Master).
SELECT	Permite executar SELECT.

SHOW DATABASES	Exibe todos os bancos de dados.
SHUTDOWN	Permite executar MySQLAdmin Shutdown.
SUPER	Permite executar CHANGE MASTER, KILL, PURGE MASTER LOGS e SET GLOBAL. Permite conectar-se ao servidor uma vez, mesmo que o max_connections tenha sido atingido.
UPDATE	Permite executar UPDATE.
USAGE	Sinônimo para "no privileges".
GRANT OPTION	Permite ao usuário repassar os seus privilégios.

Além dos privilégios, temos de definir os níveis de privilégio a serem atribuídos ao usuário.

.	Privilégio global.
db.*	Qualquer tabela do banco db.
db.tb	Apenas a tabela tb do banco de dados db. Para especificar apenas algumas colunas de uma determinada tabela, estas deverão ser listadas ao lado do privilégio (priv (colunas)). Exemplos:

```
GRANT ALL PRIVILEGES ON *.* TO Neilton@localhost IDENTIFIED BY '123456' WITH GRANT OPTION;
```

Neste caso criamos o usuário neilton, com a senha 123456 com todos os privilégios em todas as tabelas.

```
GRANT UPDATE, INSERT ON alfama.* TO neilton@localhost IDENTIFIED BY "123456";
```

Neste outro exemplo estamos criando um usuário que pode atualizar e inserir dados em todas as tabelas do banco de dados alfama, o nome do usuário é neilton e a senha é 123456.

13.2 – EXERCÍCIO PROPOSTO

1) De acordo com o exemplo abaixo explique:

GRANT ALL PRIVILEGES ON *.* TO Neilton@localhost IDENTIFIED BY '123456' WITH GRANT OPTION;

Capítulo 14 - Administrando Usuários no MySQL

14 - ADMINISTRANDO USUÁRIOS NO MYSQL

14.1 – INTRODUÇÃO

De que adianta um Sistema de Gerenciamento de Banco de Dados multiusuário, se não for possível fazer a administração de seus usuários?

Para tanto, o MySQL disponibiliza ferramentas para a gestão de usuários.

O comando `show grants` apresenta as permissões de um ou mais usuários e sua sintaxe é:

`Show grants nomeusuário.`

Para administrar o banco de dados, podemos primeiramente saber quais são os usuários do banco, para isso, primeiro temos de saber os bancos de dados disponíveis:

```
Mysql> show databases;
```

Agora vamos listar os usuários de um dos bancos de dados.

```
Mysql> use alfama;
Mysql> show grants;
```

Mostra permissão do administrador.

```
Mysql> show grants for neilton@localhost;
```

Mostra permissões do usuário neilton.

Se ainda não conhece os usuários do sistema, então deve listar os usuários disponíveis nele.

Para listar os usuários:

```
Mysql> use mysql;
Mysql> Select * from user;
```

A tabela `users` do MySQL armazena os usuários do banco de dados.

Para saber apenas os nomes:

```
Mysql> select user from user;
```

Para alterar a senha de um usuário utilize o comando `UPDATE` para a tabela `user`.

```
Mysql> UPDATE user SET Password=PASSWORD('123') WHERE user='neilton';
```

Neste caso, estamos alterando a senha do usuário neilton para 123.

Depois de alterada a senha, é necessário atualizar as regras do banco.

```
mysql> FLUSH PRIVILEGES;
```

Para remover privilégios do usuário utilize o `REVOKE`, sua sintaxe é semelhante ao `Grant`.

```
REVOKE priv [(colunas)] [, priv [(colunas)]] ...
```

ON {*. * | db.* | db.tabela}

FROM usuario [, usuario] ...

Lembre-se de que a parte **ON** do **REVOKE** deverá coincidir com a parte **ON** do **GRANT** que você deseja remover, caso isto não se verifique, o comando **REVOKE** não terá efeito algum. Além disso, o comando **REVOKE** remove apenas os privilégios do usuário, mas o usuário continuará existindo.

Exemplo:

```
mysql> revoke all on alfama.* from neilton;
```

Remove todas as permissões do banco de dados alfama, em todas as tabelas do usuário neilton.

Removendo usuários:

O comando DELETE é responsável pela remoção de usuários no MySQL. Sua sintaxe é:

DELETE FROM mysql.user WHERE user="usuario" AND host="localhost"

Exemplo:

```
DELETE FROM mysql.user WHERE user="neilton" AND host="localhost"
```

Remove o usuário neilton.

```
mysql> FLUSH PRIVILEGES;
```

Para atualizar o sistema.

14.2 – EXERCÍCIO PROPOSTO

1) Quais os comandos para listar os usuários de um banco de dados alfama?

Capítulo 15 - Backup de Banco de Dados

15 - BACKUP DE BANCO DE DADOS

15.1 – INTRODUÇÃO

Manter um banco de dados é uma das tarefas mais importantes. Entre as tarefas do Administrador de banco de dados temos a manutenção das tabelas, a atribuição e permissão de usuários e o backup. O backup é uma das tarefas mais importantes para manter o banco de dados.

O MySQL disponibiliza a ferramenta MySQLDump para fazer os backups de banco, tabelas, etc.

É possível ainda utilizar o MySQL Workbench ou o PHPMyAdmin para efetuar backups, mas veremos aqui através de comando.

O MySQLDump é uma ferramenta que possibilita copiar bancos de dados dentro do Sistema de Gerenciamento de Bancos de Dados MySQL, funciona de forma simples, colocando o conteúdo e estrutura das tabelas em um arquivo texto, é o que chamamos de DUMP.

O MySQLDump pode produzir arquivos no formato SQL que contenham declarações CREATE TABLE e INSERT para recriar os arquivos ou para produzir arquivos de dados delimitados por tabulações.

Passos para efetuar um backup:

- O primeiro passo indicado é criar um novo log antes do backup, então dentro do MySQL rode o comando flush log:

```
Mysql> flush logs;
```

- Saia do MySQL (Exit) e rode o comando MySQLDump.

Sintaxe:

```
mysqldump -u <usuario> -p<senha> -a > mysql.sql
```

A opção -a indica para fazer backup de todos os bancos de dados do sistema.

Para efetuar o backup de um banco de dados específico é necessário indicar o banco de dados, conforme o exemplo abaixo:

```
Mysqldump -u root -p 123456 --databases alfama > alfama.sql
```

O comando faz o backup do banco de dados alfama, gerando o arquivo alfama.sql.

Em caso de problemas com o banco de dados, para restaurar o backup faça o seguinte:

```
mysql -u root -p 123456 < alfama.sql
```

O exemplo acima restaura o banco de dados alfama através do backup efetuado no arquivo .sql.

15.2 – EXERCÍCIO PROPOSTO

1) Para que serve a ferramenta MySQLDump?

Capítulo 16 - Chaves

16 - CHAVES

16.1 – INTRODUÇÃO

Entende-se por chaves como o conceito básico utilizado para identificar linhas e estabelecer uma relação entre linhas da tabela de um banco de dados relacional. Nos bancos de dados relacionais há pelo menos dois tipos de chaves: Chave Primária e a Chave Estrangeira.

Na verdade, as chaves são restrições de integridade impostas ao banco de dados.

• Chave Primária:

Identifica cada registro dando-lhe unicidade. A chave primária nunca se repete dentro de uma estrutura de tabela, a exemplo, temos o CPF, que é uma chave primária de todo cidadão brasileiro, nunca se repete. Pode ser formada por um único campo ou por vários campos de uma tabela.

A chave primária é uma coluna ou conjunto de colunas, cujos valores distinguem uma linha das demais dentro de uma mesma tabela.

Quando a chave primária é formada por vários campos, pode-se dizer que é uma chave primária composta. Uma tabela não pode ter mais de uma chave primária.

• Chave Estrangeira:

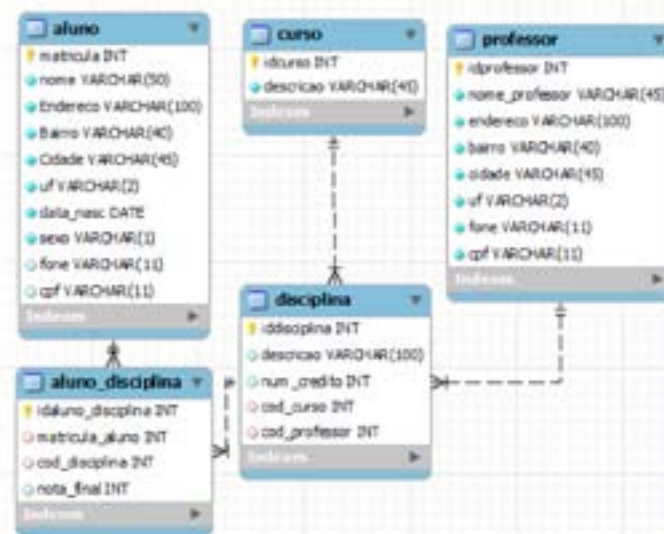
A chave estrangeira corresponde aos campos ou campo de uma tabela, cujos valores cadastrados aparecem necessariamente na chave primária de outra tabela.

A chave estrangeira é o mecanismo que vai permitir ao SGBD de implementar um relacionamento entre as tabelas, daí o nome Banco de Dados Relacional.

Uma chave estrangeira impõe algumas restrições ao banco de dados:

- Quando há uma inclusão de uma linha na tabela que contém a chave estrangeira, é necessário que o valor a ser inserido no campo chave estrangeira exista na tabela principal.
- Quando for alterado o valor da chave estrangeira, é necessário que o novo valor cadastrado exista na chave primária da tabela principal.
- Quando for removido um dado da tabela principal, cujo valor exista em alguma tabela relacionada, a exclusão deve ser cancelada, ou o registro na tabela relacionada também deverá ser excluído.

Exemplo de modelagem de dados utilizando chaves.



16.2 – EXERCÍCIO PROPOSTO

1) Conceitue Chave Primária e Chave Estrangeira.

Capítulo 17 - Cardinalidade

17 - CARDINALIDADE

17.1 – INTRODUÇÃO

As tabelas de um banco de dados, como vimos no capítulo anterior, interagem entre si através de relacionamento existente entre as informações.

A interação desenvolvida entre as tabelas possui cardinalidade.

Existem diversos tipos de cardinalidade, entre elas temos:

- **Cardinalidade Mínima:** a cardinalidade mínima define se o relacionamento entre duas tabelas é obrigatório ou não.

Exemplo: Um país pode ou não possuir UF. Cada UF pode pertencer a um único país.

- **Cardinalidade Máxima:** a cardinalidade máxima define a quantidade máxima de ocorrências entre as entidades que participam de um relacionamento.

- **Cardinalidade Um para Um:** na cardinalidade um para um, o relacionamento entre as tabelas sempre ocorre de um para um, ou seja, para cada registro de uma tabela, só existe uma única ocorrência relacionada em outra tabela.

Exemplo: Um estado tem uma única capital e esta capital pertence a um único estado.

- **Cardinalidade Um para N:** na cardinalidade um para n é possível que um registro de uma determinada tabela se relacione com vários registros de outra tabela.

Exemplo: Um país possui vários estados.

- **Cardinalidade N para N:** a cardinalidade n para n representa um modelo em que vários registros de uma tabela se relacionam com vários registros de outra tabela.

Exemplo: Vários alunos podem estar matriculados em várias matérias.

A cardinalidade que é outra forma de integridade de dados, como vimos, é possível criar limitações entre os campos chaves para que os dados tenham regras para serem armazenadas, desta forma criando uma integridade maior entre as tabelas, gerando maior segurança na informação e maior confiabilidade dos dados, buscando atender a um dos pilares da norma NBR 27002 que é confiabilidade.

17.2 – EXERCÍCIO PROPOSTO

1) Explique os tipos de cardinalidade 1:1, 1:N e N:N.

Capítulo 18 - Regras de Integridade de Dados no MySQL

18 - REGRAS DE INTEGRIDADE DE DADOS NO MYSQL

18.1 – INTRODUÇÃO

A integridade é um fator crucial para o sucesso de um banco de dados bem projetado. As regras de integridade são importantes para que um banco de dados evite redundância de dados e garanta a integridade.

A obrigatoriedade de informação dos dados de uma tabela é uma regra de integridade (por exemplo: deixar um campo como not null).

18.2 - IMPLEMENTANDO CHAVE PRIMÁRIA

As chaves primárias e estrangeiras são outras regras de integridade para as tabelas.

Sintaxe:

Primary key(atributo1, atributo2 ... Atributox);

Check(P);

Foreign key (atributo1, atributo2 .. Atributo x) references nome da tabela.

O Primary Key diz que o atributo1, atributo2 ... e atributo x formam a chave primária da tabela. A chave primária corresponde ao campo, ou campos, que não podem de forma alguma ter valores repetidos dentro da tabela.

O Check impõe um domínio a um determinado campo da tabela, forçando-o a aceitar apenas as opções pertencentes ao domínio estabelecido. Por exemplo, para a coluna Sexo pode ser estabelecido o domínio ('M','F'). Isto é implementado através da seguinte cláusula check, que ao se implementar uma tabela check Sexo in ('M','F'), desta forma o MySQL fará a verificação sempre que um valor for inserido.

O Foreign Key é a chave estrangeira, ou seja, inclui a relação de atributos que constituem a chave estrangeira (atributo 1, 2 .. X) e em quais tabelas esses campos fazem referência aos campos formadores de chaves primárias.

- Exemplo de criação de tabelas com chave primária:

```
Create table aluno(
Matricula integer auto_increment,
Nome varchar(50) not null,
Fone varchar(11) not null,
Primary key(matricula)
);
```

- Exemplo de criação de tabelas com chave estrangeira:

```
Create table turma(
Id integer auto_increment,
Matricula_aluno integer,
Primary key(id),
Foreign key(matricula_aluno) references aluno(matricula) );
```

18.3 – EXERCÍCIO PROPOSTO

1) O que se coloca em um campo de uma tabela para que se tenha uma regra de integridade de obrigatoriedade de informação dos dados?

Capítulo 19 - Manipulando Dados em Tabelas Relacionadas

19 - MANIPULANDO DADOS EM TABELAS RELACIONADAS

19.1 – INTRODUÇÃO

O relacionamento cria uma interdependência entre os campos de tabelas distintas, isto faz com que as tabelas tenham integração.

Com o relacionamento, temos de pensar na forma de se inserir dados em uma tabela.

No exemplo do capítulo anterior, criamos uma tabela-aluno e uma tabela-turma.

A tabela-turma possui como chave estrangeira os alunos da turma, o que implica em ter primeiro o cadastro do aluno, caso contrário, não será possível fazer o cadastro da turma.

Isto indica integridade dos dados. O mesmo acontece com estados em um país, em que para cadastrar um estado, o correto é que ele esteja ligado a um país e não perdido no banco, para tanto é necessário ligação e itens como o not null.

Por isso é importante o projeto do banco de dados pensar nas ligações e itens que criem integridade.

Outro ponto importante é que não é possível remover o país, por exemplo, sem remover os estados, ou ao mesmo tempo em que eu removo os países, os estados serão removidos automaticamente.

Por isso que temos uma integridade dos dados. Quando temos esta referência entre as tabelas, os dados são interligados, não possibilitando que se percam dados acidentalmente, já que para remover algumas informações, o Administrador deve remover uma árvore de informações. Isto cria uma maior confiabilidade dos dados do banco além de maior integridade, apontando para uma maior integração com a norma NBR 27002, que em conjunto com o backup utilizando-se o dump, temos todos os pilares da segurança atendidos dentro do MySQL: Confiabilidade, Integridade e Disponibilidade.

Outro ponto importante da integridade é a forma de se inserir informações, pois como no exemplo, para inserir um estado devemos primeiro ter um país, para cadastrar um município devemos primeiro ter um estado, desta forma, obriga o desenvolvedor ou o usuário do banco de dados, que insira os dados de forma correta. É importante que durante a modelagem o DBA aplique conceitos de normalização, integridade, etc., fazendo com que as tabelas sejam organizadas e tenham todas as informações necessárias ao sistema.

19.2 – EXERCÍCIO PROPOSTO

1) De acordo com os relacionamentos entre tabelas temos:

Uma tabela-aluno possui como chave estrangeira a turma onde estará matriculado, o que implica em ter primeiro o cadastro da turma. É possível fazer o cadastro do aluno sem primeiro ter feito o cadastro da turma onde estará matriculado?

Capítulo 20 - Seleção de Dados

20 - SELEÇÃO DE DADOS

20.1 – INTRODUÇÃO

De que adianta armazenar os dados e possuir integridade se não for possível selecionar os dados do SGBD

A seleção de dados é uma tarefa de extrema importância, tanto para o Administrador de banco de dados, como para o Programador. Do que adianta um programa que não resgata informação?

Relembrando, o comando que seleciona dados em uma tabela é o select, que pode ser utilizado para selecionar uma ou várias colunas em uma tabela de banco de dados.

Ex.: select * from alunos;

Seleciona todos os dados, exibindo todos os campos da tabela-alunos.

Sintaxe completa do select:

Select [colunas] from [tabela] where [condição] group by [expressão] Having [condição] order by[expressão]

Colunas: pode ser indicada uma ou mais colunas utilizando (,) ou o (*) para indicar todas as colunas.

Exemplo:

Select matricula, nome from aluno

Seleciona os campos matrícula e nome da tabela-aluno.

A cláusula where é responsável por restringir quais linhas na tabela serão apresentadas.

Operadores e seus significados:

= --> igual	>= --> maior ou igual
> --> maior	<= --> menor ou igual
< --> menor	AND --> e lógico
<> --> diferente	OR --> ou lógico

Exemplo:

Select * from alunos where sexo='m'

Seleciona todos os alunos que o sexo seja igual a m.

Select * from alunos where matricula=10

Seleciona o aluno que a matrícula seja igual a 10.

Select * from alunos where matricula>10 AND matricula<20

Seleciona o aluno que a matrícula seja maior que 10 e menor que 20.

Nota-se que com o where e utilizando os operadores é possível delimitar os dados que serão apresentados ao usuário, tornando a velocidade do resultado maior, evitando inundar a rede com informações inúteis e que não serão utilizadas. Quanto menor o número de informações, mais ágil é o resultado para o usuário. Buscar 1000 usuários e buscar 1 usuário é diferente, e trafegar pela rede com 10kb de informação ou 1gb de informação também é diferente, então ao fazer as consultas, evite trazer mais informações do que você necessita.

20.2 – EXERCÍCIO PROPOSTO

1) Explique o exemplo:

Select * from alunos where matricula>10 AND matricula<20

Capítulo 21 - Seleção: ordenação, agrupamento

21 - SELEÇÃO: ORDENAÇÃO, AGRUPAMENTO

21.1 – INTRODUÇÃO

A seleção de grandes informações às vezes é necessária, mas é preciso organizar estas informações e agrupá-las por tipo, desta forma, ao selecionar dados em uma tabela, além de ter que limitar a quantidade de dados, pode ser necessário organizar e agrupar os dados.

A ordenação de informações é feita da seguinte forma: escolhemos o campo que será a base da ordenação e mandamos ordenar. Imagine que queira uma lista de usuários em ordem alfabética, logo mandamos ordenar por nome, mas se quiser ordenar por idade do mais novo ao mais velho, mandamos ordenar por data de nascimento de forma decrescente. Para ordenar utilize:

• Order by

A cláusula order by serve para ordenar os dados que serão apresentados através do comando select. A ordem de apresentação pode ser crescente (ASC) ou decrescente (DESC).

Se não for informada a ordem de apresentação, será feito o padrão crescente.
Sintaxe do comando:

Order by coluna1, coluna2 ... DESC/ASC;

Exemplo:

Select * from alunos order by matricula;

Select * from alunos where matricula > 10 and matricula < 20 order by matricula desc;

O agrupamento de informações às vezes é necessário, imagine que você precise listar usuários organizados por estados? Para isso, podemos utilizar o agrupamento de informações pelo campo estado. Outro exemplo seria agrupar por sexo.
Para agrupar utilize:

• Group by

A cláusula group by agrupa em uma única linha todas as linhas selecionadas que compartilham o mesmo valor de agrupamento.

Um item do group by pode ser o nome de uma coluna ou o número da coluna na estrutura da tabela.

Sintaxe do group by:

Group by coluna1, coluna2, coluna

Exemplo:

Select * from alunos group by estado;

O resultado será a apresentação dos alunos agrupados por estado.

21.2 – EXERCÍCIO PROPOSTO

1) Defina a sintaxe do comando para obter o resultado de uma apresentação dos alunos agrupados por estado.

Capítulo 22 - Funções de Agregação

22 - FUNÇÕES DE AGREGAÇÃO

22.1 – INTRODUÇÃO

Uma função de agregação serve para retornar um único resultado para várias linhas de uma tabela. Por exemplo: saber quantos alunos têm em um curso, a média das notas dos alunos, o valor máximo e mínimo, etc.

Uma função de agregação recebe os dados de várias linhas de uma tabela, porém um único valor de retorno é apresentado.

Principais funções de agregação:

Count --> retorna o número de ocorrências de uma coluna na tabela.

Sum --> retorna a soma dos valores de uma coluna.

Avg --> retorna a média de valores da coluna.

Max --> retorna o maior valor da coluna.

Min --> retorna o menor valor da coluna.

Distinct --> não permite que valores iguais de uma coluna sejam apresentados.

A função de agregação sempre está associada a uma coluna da tabela, desta forma a sintaxe do comando é:

Função_de_agregação(coluna)

Exemplo de função de agregação:

Select max(nota_aluno) as maior_nota from aluno;

A maior nota de todos os alunos, a opção as, indica o nome que aparecerá para exibir a informação.

Select min(nota_aluno) as menor_nota from aluno;

Seleciona a menor nota de todos os alunos.

Select avg(nota_aluno) as media from aluno;

Seleciona as notas e exibe a média de todas as notas na tela.

Select count(nome) as quantidade_alunos from aluno;

Exibe quantos alunos têm cadastrados na tabela-aluno.

Select sum(mensalidade_alunos) as valor_total from alunos;

Exibe o resultado da soma dos valores do campo mensalidade_alunos.

Select distinct(uf) from alunos;

Lista apenas as uf's únicas dos alunos, sem repetição.

O distinct pode ser utilizado quando há dados repetidos e o DBA precisa trazer uma lista única. No exemplo acima, imagine que tenha usuários de SP, SE, BA, e se selecionarmos só os estados, o sistema vai retornar várias vezes a mesma informação. Com o distinct ele traz apenas uma vez a informação solicitada, tornando o retorno menor.

22.2 – EXERCÍCIO PROPOSTO

1) Explique o exemplo:

Select max(nota_aluno) as maior_nota from aluno;

Capítulo 23 - Junção

23 - JUNÇÃO

23.1 – INTRODUÇÃO

Nos modelos de banco de dados, principalmente nos que possuem um número elevado de tabelas, às vezes é necessário elaborar uma consulta em diversas tabelas ao mesmo tempo.

A operação de seleção necessita que o especialista defina os critérios de agrupamento das tabelas.

Os critérios para o agrupamento são chamados de junção.

A junção de tabelas apresenta como resultado uma tabela derivada de duas ou mais tabelas de acordo com a regra elaborada.

As regras de junção são semelhantes às regras da teoria de conjunto da matemática, como união, intersecção e diferença.

Dependendo da condição da junção é possível obter os seguintes tipos de junção:

- **Produto Cartesiano:** retorna toda a combinação de registro de duas tabelas de dados, uma vez que não existe nenhuma restrição definida na cláusula where.
- **Junção Interna (inner-join):** a consulta retornará apenas os registros que atendem às condições definidas nas tabelas de dados.
- **Junção Externa (outer-join):** semelhante à junção simples, porém retornará também os registros da fonte de dados externa que não estão presentes na tabela interna.

Na definição da junção são necessários alguns cuidados como:

- Identificar as chaves primárias das tabelas.
- Identificar as chaves estrangeiras das tabelas.
- Identificar os relacionamentos entre as tabelas.

Mas então, o que é a Junção?

A junção é uma seleção entre duas ou mais tabelas, sendo exibida como uma tabela única.

Imagine que queira selecionar dados das tabelas aluno, turma, professor, mas exibir todos os alunos da matéria de Banco de Dados, lecionados pelo Professor Neilton?

A junção é utilizada para trazer este resultado. Ao invés de você fazer uma seleção em cada tabela, você faz uma junção na seleção, trazendo diversos dados ao mesmo tempo.

Este recurso está disponível em bancos como Oracle, PostgreSQL, SQL Server e também no MySQL.

O único ponto a ser considerado para a seleção por junção é a velocidade, verifique o desempenho da sua seleção antes de concluir.

23.2 – EXERCÍCIO PROPOSTO

1) Quais os tipos de Junção? Especifique cada uma.

Capítulo 24 - Junção na Prática

24 - JUNÇÃO NA PRÁTICA

Vamos iniciar a Junção Cartesiana.

24.1 - JUNÇÃO PRODUTO CARTESIANO

```
Select cidade.codigo, cidade.nome, sub.bairro from cidade, sub where cidade.codigo = sub.codigo_cidade
```

Neste exemplo, selecionamos na tabela-cidade e na tabela-sub os dados, quando o código sub da cidade (chave estrangeira) for igual ao código da cidade (chave primária).

24.2 - JUNÇÃO INTERNA/INNER JOIN

O que é Junção Interna?

A junção interna utiliza a cláusula inner para a função de junção, apesar de não ser obrigatória. Pode parecer à primeira vista, que a junção interna é igual à junção cartesiana, mas as junções internas sempre precisam da condição da junção, sendo obrigatória a cláusula ON.

```
Select uf.nome, pais.nome from uf inner join pais on uf.codigo_pais = pais.codigo
```

Neste exemplo, foi selecionado na tabela-estado o nome do estado e na tabela-país o nome do país, quando o estado tiver o código do país (chave estrangeira) igual ao código do país da tabela país (chave primária).

24.3 - JUNÇÃO EXTERNA

A seleção externa utiliza a cláusula outer, mas esta também não é obrigatória, o que realmente caracteriza as junções externas são as cláusulas left e right.

24.4 - LEFT OUTER JOIN

Executa uma junção interna.

Verifica quais linhas da primeira tabela não possuem relacionamento com a segunda tabela.

Para cada linha da segunda tabela que não possui relacionamento é inserido um campo com o valor null.

```
Select * from pais left outer join uf ON uf.codigo_pais = pais.codigo;
```

24.5 - RIGHT OUTER JOIN

Executa uma junção interna.

Verifica quais linhas da segunda tabela não possuem relacionamento com a primeira tabela.

Para cada linha da primeira tabela que não possuir relacionamento é inserido um campo com o valor null.

```
Select * from pais left outer join uf ON uf.codigo_pais = pais.codigo;
```

24.6 – EXERCÍCIO PROPOSTO

1) Qual o resultado do código abaixo:

```
Select cidade.codigo, cidade.nome, sub.bairro from cidade, sub where cidade.codigo =  
sub.codigo_cidade
```

REFERÊNCIAS

Documentação Oficial MySQL - Características Principais. Disponível em:

<http://dev.mysql.com/doc/refman/4.1/pt/features.html>. Acessado em: 05/08/2012.

Documentação Oficial MySQL - Exemplos de Banco de Dados. Disponível em:

<http://dev.mysql.com/doc/index-other.html>. Acessado em: 05/08/2012.

Documentação Oficial MySQL - Funções para Data e Hora. Disponível em:

<http://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html>. Acessado em: 05/08/2012.

Documentação Oficial MySQL - Tipos de Campos. Disponível em:

<http://dev.mysql.com/doc/refman/4.1/pt/column-types.html>. Acessado em: 05/08/2012.

Ferramentas para Administração do Banco de Dados MySQL. Disponível em:

<http://www.w3avenue.com/2009/12/03/free-tools-for-mysql-design-and-administration/>. Acessado em: 05/08/2012.

MANZANO, José Augusto N. G. MySQL 5.5. - *Interativo*. São Paulo: Erica, 2011.

MILANI, André. MySQL - *Guia do Programador*. São Paulo: Novatec, 2007.

RANGEL, Alexandre. MySQL - *Projeto, Modelagem e Desenvolvimento*. Rio de Janeiro: Altabook, 2005.

TONSIG, Sérgio Luiz. MySQL - *Aprendendo na Prática*. São Paulo: Ciência Moderna, 2006.



alfamacursos.com.br